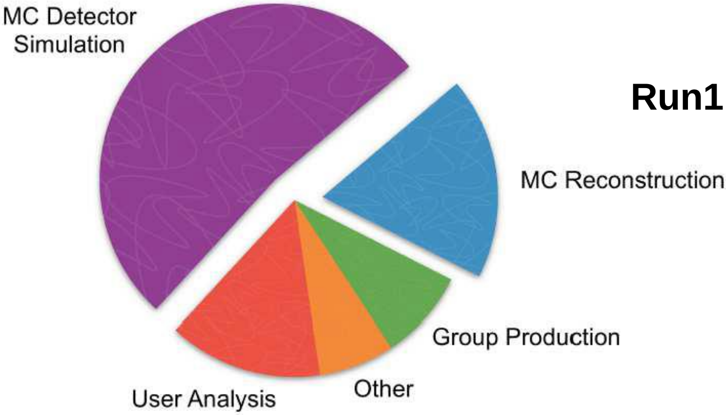
GeantV

TAKING UP THE TECHNOLOGY CHALLENGE

CERN openlab Open Day
June 10, 2015

DETECTOR SIMULATION AND THE LHC CHALLENGE

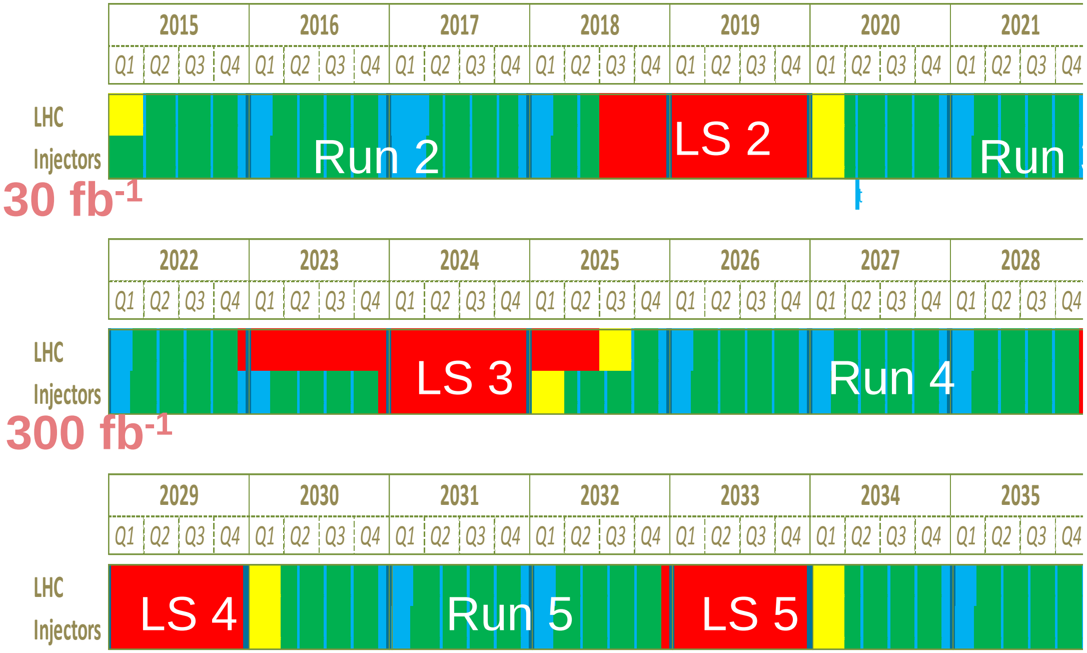
ATLAS GRID CPU utilization



Run1➔Run2

LHCb GRID usage	**2013**	**2016**
Sim	**64.5%**	**63%**
User	20.2%	8%
Rest (str, repro, rec)	15.3%	29%

O. Bruning 2014 – ECFA HL LHC workshop



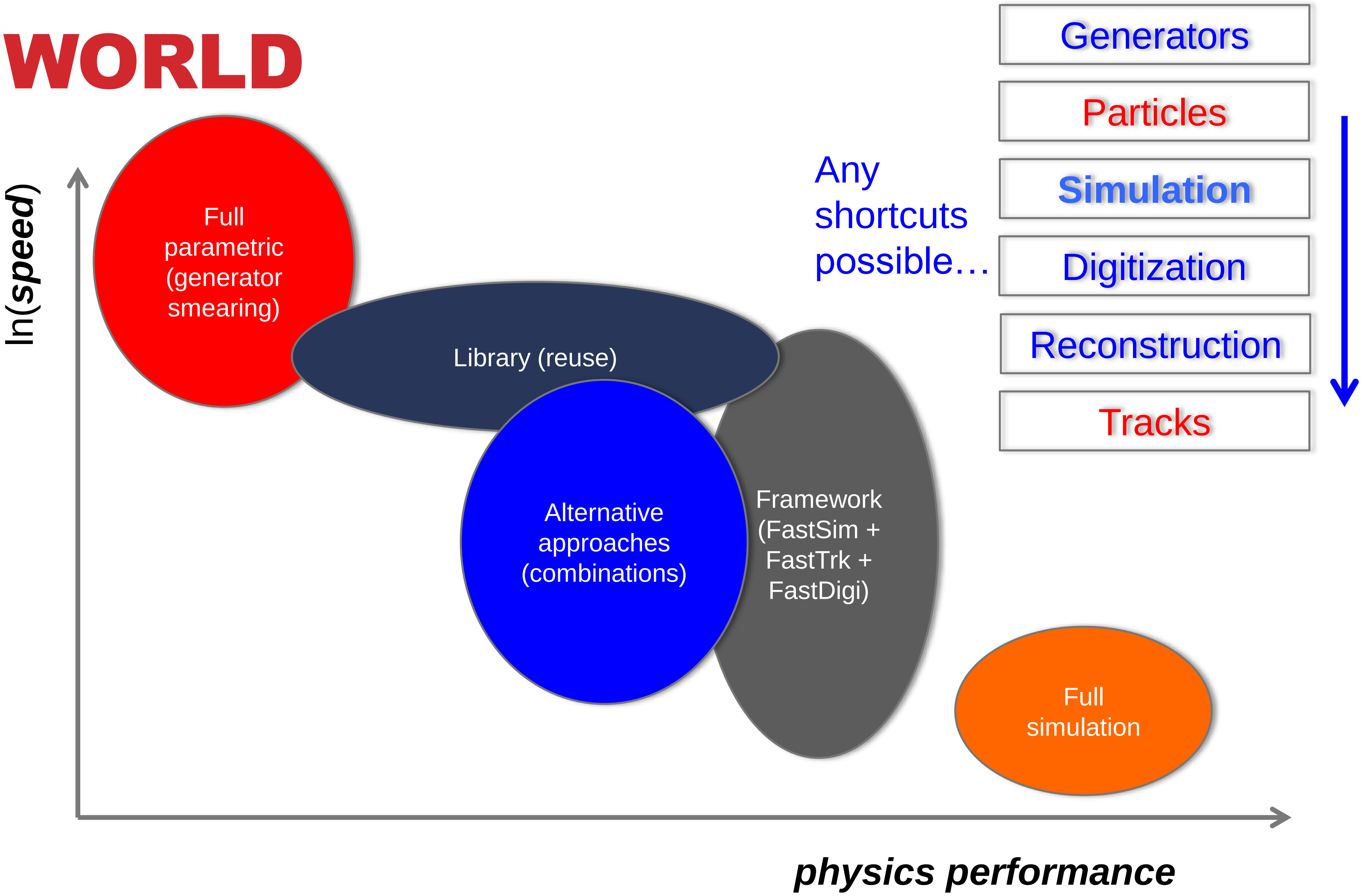
Goal of 3'000 fb^{-1} by mid 2030's

Aiming for a factor of 100 integrated luminosity over 20 years

- **x25 data rate vs. Run1, with much increased pile-up**
  - Simulated sample requests will follow in some way…
  - **More fast simulation…**
- **Detector studies for upgrades, new experiments**
  - **… and faster full simulation**

Factors needed in throughput…

HEP SIMULATION WORLD



Full parametric (generator smearing)

Library (reuse)

Alternative approaches (combinations)

Framework (FastSim + FastTrk + FastDigi)

Full simulation

Any shortcuts possible…

Generators

Particles

Simulation

Digitization

Reconstruction

Tracks

COMPUTING TECHNOLOGY CHALLENGES

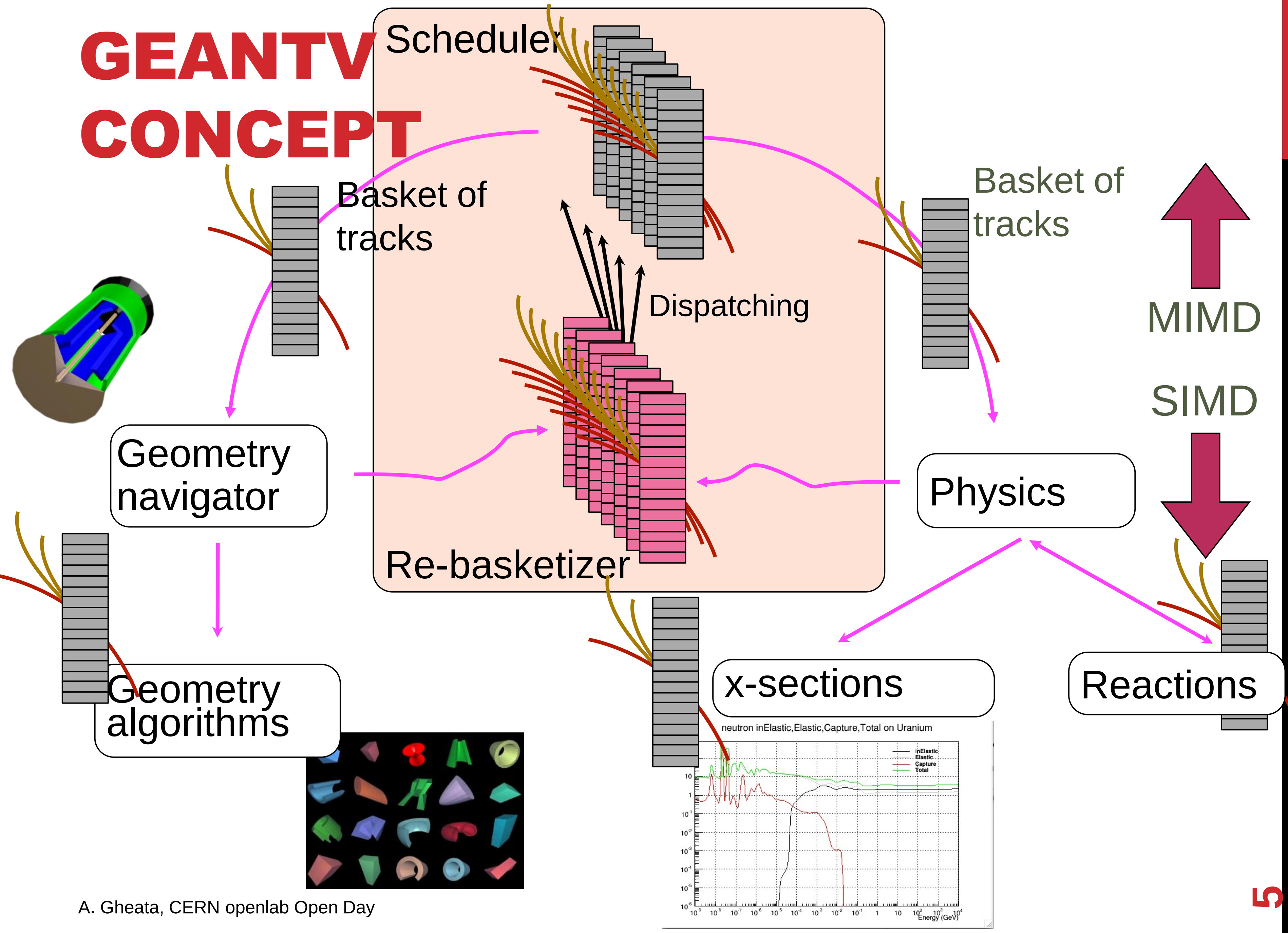
Making the software parallel ≠ gaining throughput

- **The last requires re-thinking the code in terms of data and code locality, better use of SIMD vectorisation**
  - Less CPI and much less cache misses
  - Rethinking algorithms and processing flow
  - Using standard or opportunistic resources efficiently

HEP software hibernating for long into “sequentiality” waking up in a “parallel” world

- **Multi-stage process: make the first steps, interact with the environment, change old behaviors, live with diversity…**

GEANTV CONCEPT



Scheduler

Basket of tracks

Basket of tracks

Dispatching

Re-basketizer

MIMD

SIMD

Geometry navigator

Geometry algorithms

Physics

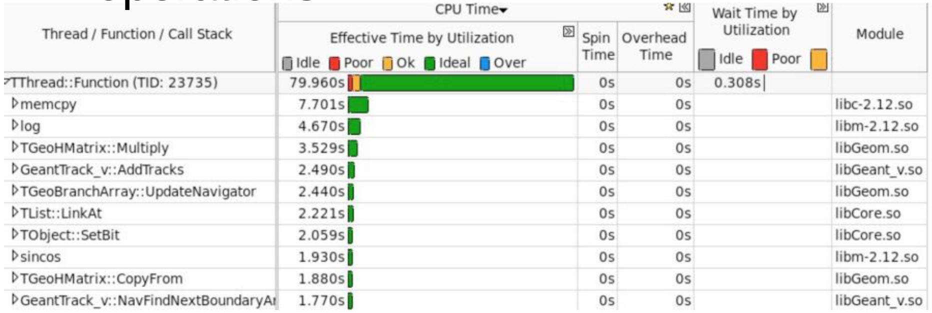
x-sections

Reactions

A. Gheata, CERN openlab Open Day

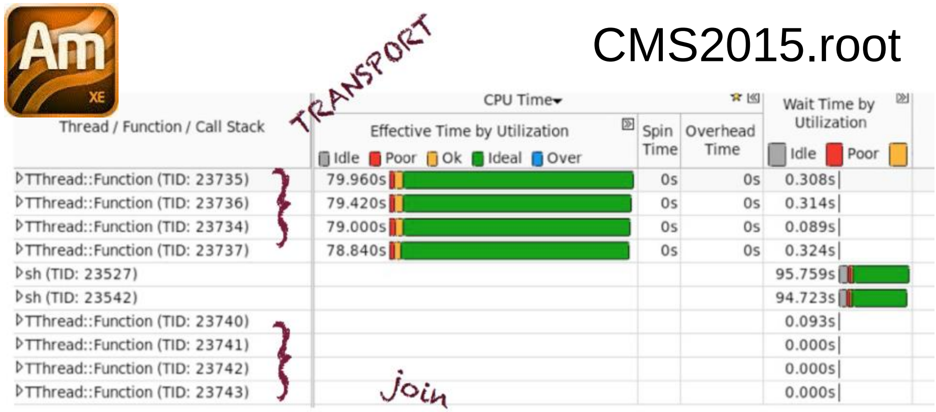
FINE GRAIN PARALLELISM IS CHALLENGING

- Several parameters to be tuned
- Performance is monitored
  - Allows detecting and fixing bottlenecks
- Amdahl still high due to rebasketizing operations

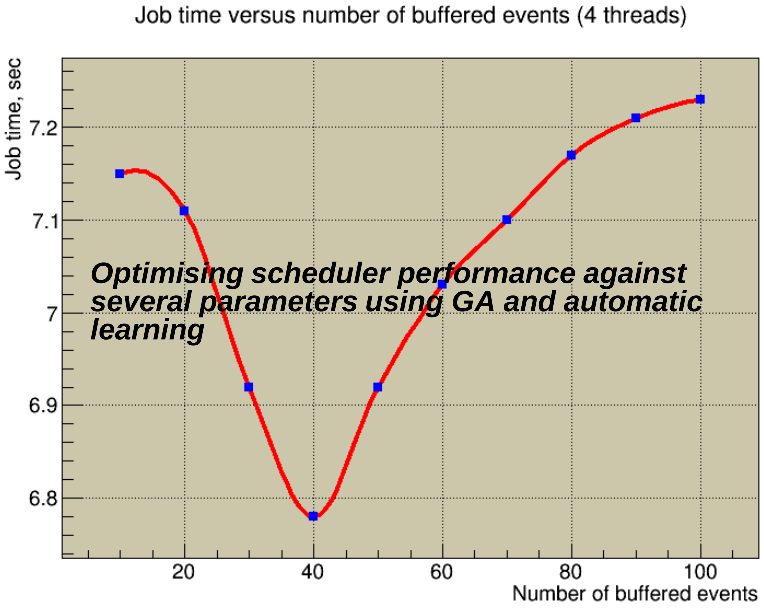


Thread / Function / Call Stack	CPU Time			Wait Time by Utilization	Module
	Effective Time by Utilization Idle Poor Ok Ideal Over	Spin Time	Overhead Time	Idle Poor	
TThread::Function (TID: 23735)	79.960s	0s	0s	0.308s	
memcpy	7.701s	0s	0s		libc-2.12.so
log	4.670s	0s	0s		libm-2.12.so
TGeoHMatrix::Multiply	3.529s	0s	0s		libGeom.so
GeantTrack_v::AddTracks	2.490s	0s	0s		libGeant_v.so
TGeoBranchArray::UpdateNavigator	2.440s	0s	0s		libGeom.so
TList::LinkAt	2.221s	0s	0s		libCore.so
TObject::SetBit	2.059s	0s	0s		libCore.so
sincos	1.930s	0s	0s		libm-2.12.so
TGeoHMatrix::CopyFrom	1.880s	0s	0s		libGeom.so
GeantTrack_v::NavFindNextBoundaryAı	1.770s	0s	0s		libGeant_v.so

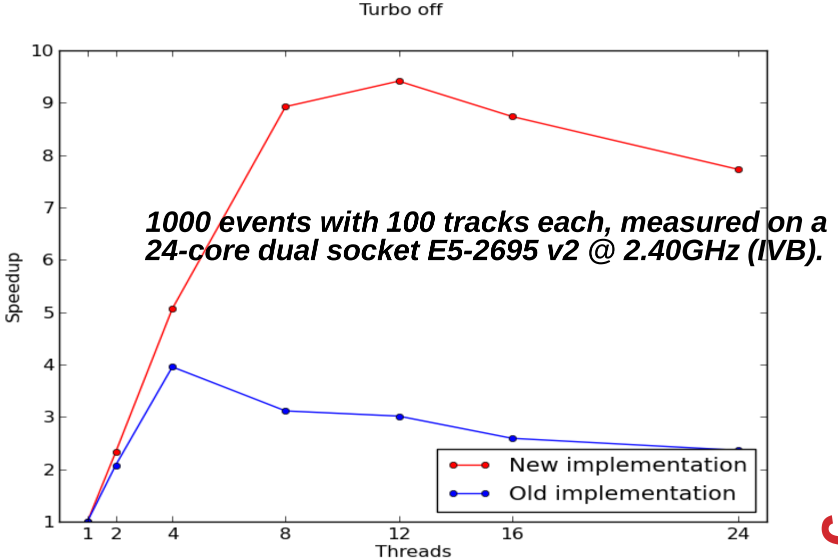
CMS2015.root



Thread / Function / Call Stack	CPU Time			Wait Time by Utilization
	Effective Time by Utilization Idle Poor Ok Ideal Over	Spin Time	Overhead Time	Idle Poor
TThread::Function (TID: 23735)	79.960s	0s	0s	0.308s
TThread::Function (TID: 23736)	79.420s	0s	0s	0.314s
TThread::Function (TID: 23734)	79.000s	0s	0s	0.089s
TThread::Function (TID: 23737)	78.840s	0s	0s	0.324s
sh (TID: 23527)				95.759s
sh (TID: 23542)				94.723s
TThread::Function (TID: 23740)				0.093s
TThread::Function (TID: 23741)				0.000s
TThread::Function (TID: 23742)				0.000s
TThread::Function (TID: 23743)				0.000s



Optimising scheduler performance against several parameters using GA and automatic learning

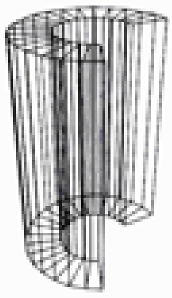


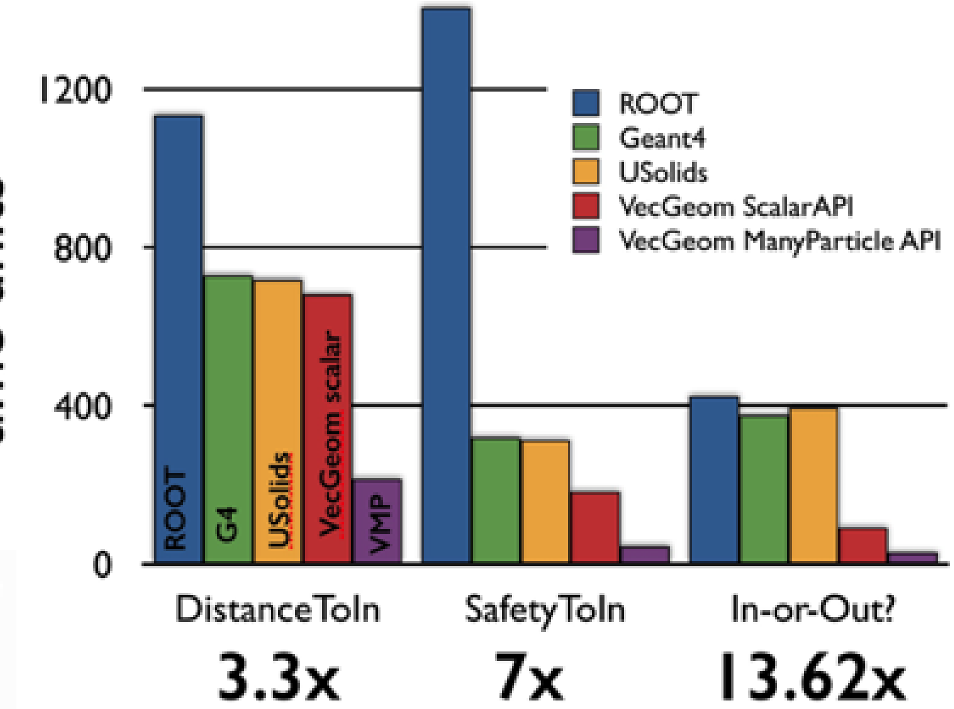
1000 events with 100 tracks each, measured on a 24-core dual socket E5-2695 v2 @ 2.40GHz (IVB).

GEOMETRY

We have developed a library of vectorised geometry algorithms to take maximum advantage of SIMD architectures

We obtain excellent performance gains also in scalar mode





The code is available in the AIDA usolid library and is being validated for Geant4 use

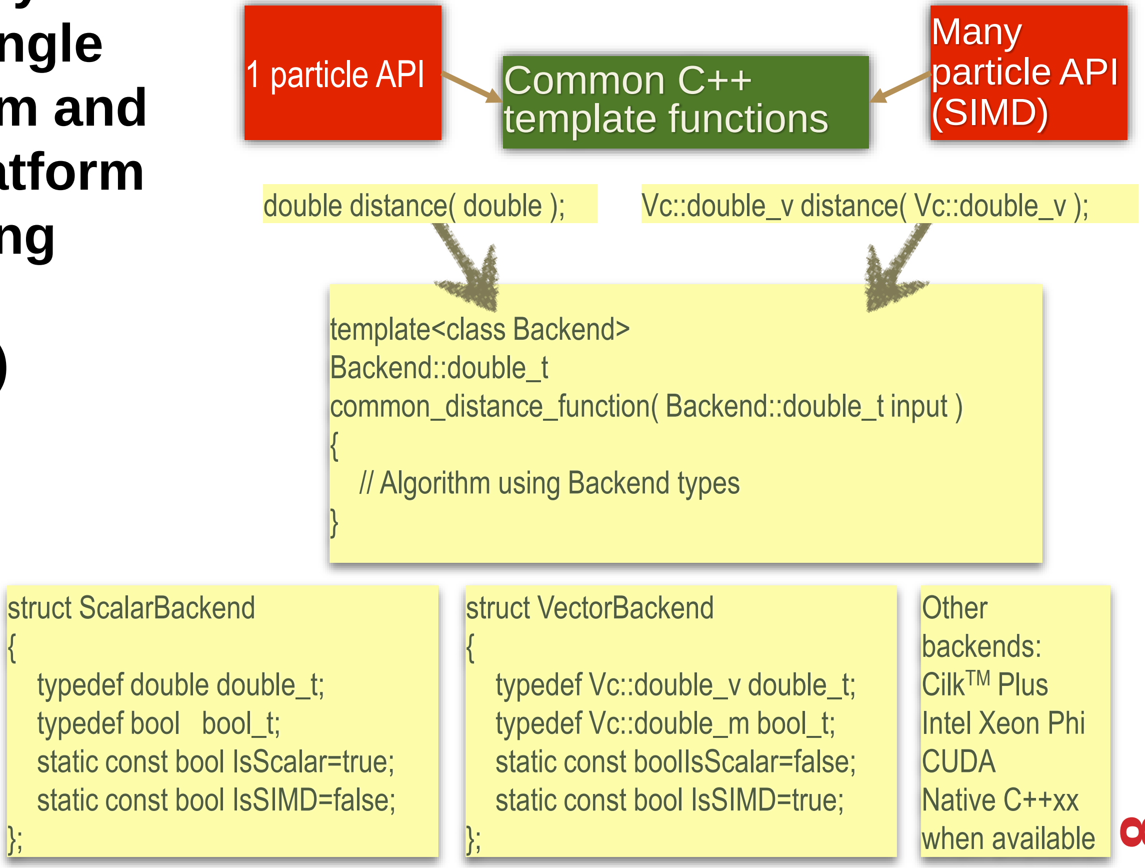
PORTABILITY

http://code.compeng.uni-frankfurt.de/projects/vc

Long-term maintainability of the code => write one single version of each algorithm and to specialise it to the platform via template programming and low level optimised libraries (Vc in our case)

Results are quite encouraging: may be portable HPC is NOT an oxymoron after all…

“Backend” is a (trait) struct encapsulating standard types/properties for “scalar, vector, CUDA” programming; makes information injection into template function easy



```
double distance( double );
```

```
Vc::double_v distance( Vc::double_v );
```

```
template<class Backend>
Backend::double_t
common_distance_function( Backend::double_t input )
{
   // Algorithm using Backend types
}
```

```
struct ScalarBackend
{
   typedef double double_t;
   typedef bool   bool_t;
   static const bool IsScalar=true;
   static const bool IsSIMD=false;
};
```

```
struct VectorBackend
{
   typedef Vc::double_v double_t;
   typedef Vc::double_m bool_t;
   static const boolIsScalar=false;
   static const bool IsSIMD=true;
};
```

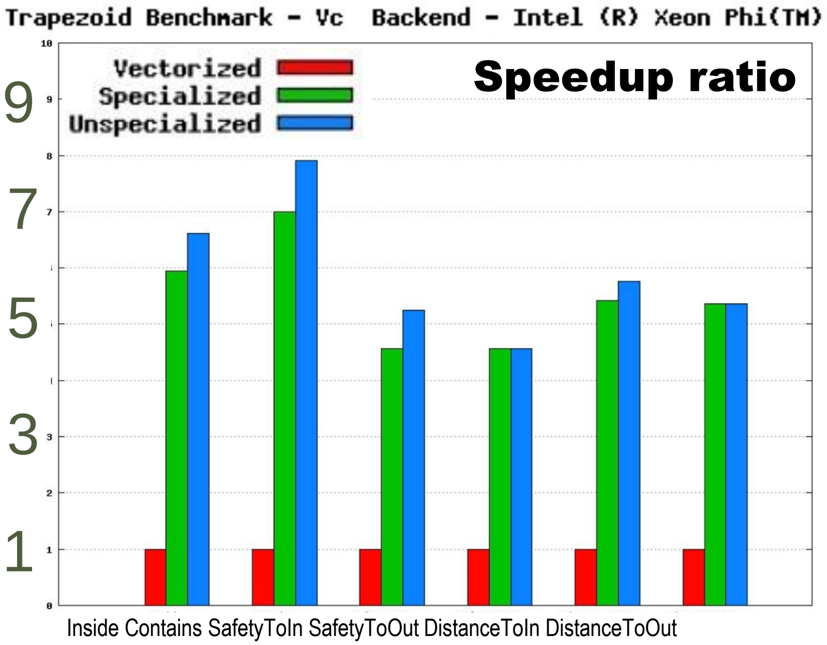
Other backends:
Cilk^{TM} Plus
Intel Xeon Phi
CUDA
Native C++xx when available

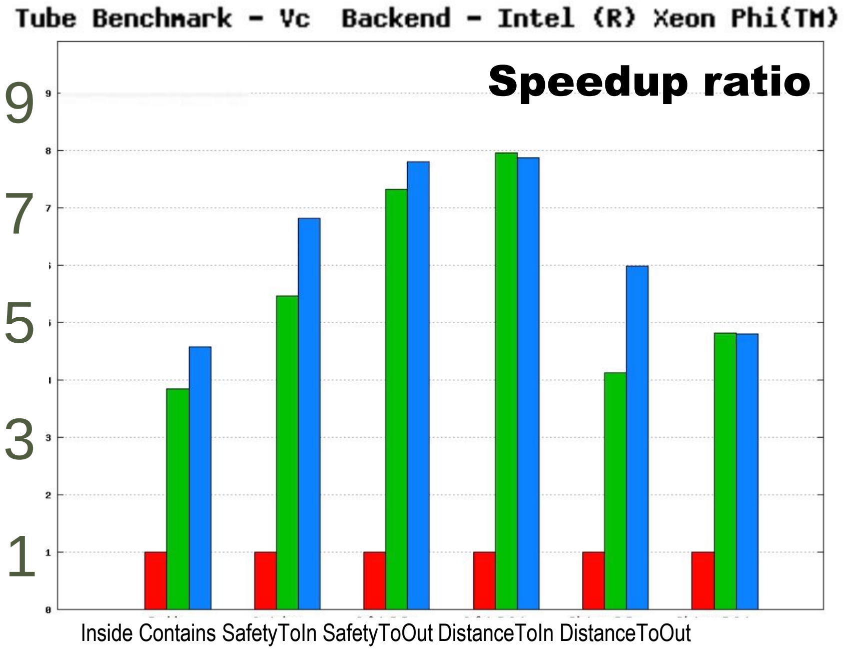
HOW DOES IT WORK?

Results obtained by CERN and UNESP Intel IPCCs on Xeon Phi

SIMD optimisation gives us more than half order of magnitude

Remember: for single thread SIMD the max lim is 8 (IMCI) – difficult to do better…





https://software.intel.com/en-us/articles/ipcc-at-cern-european-organisation-for-nuclear-research

BASKET (NO-)OVERHEAD

Full CMS2015 geometry with 1MeV cuts

Comparison (tabulated physics) of

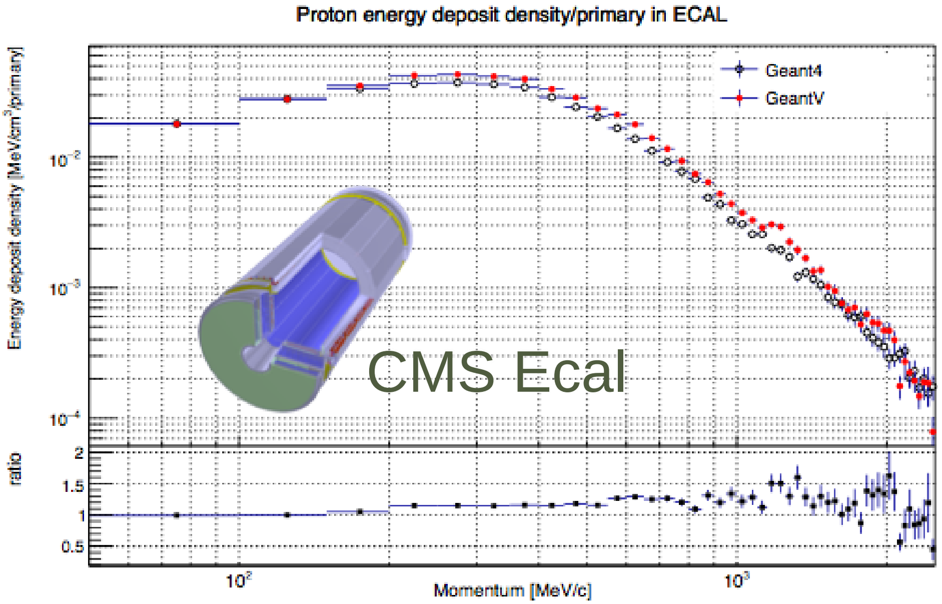
- **GeantV scheduler w. TGeo (the geom package of ROOT) in single thread**
- **Geant4**

NOT a performance comparison!!

- **Cannot compare functionality of the two programmes**

But a circumstantial proof that the overhead of basket handling is under control

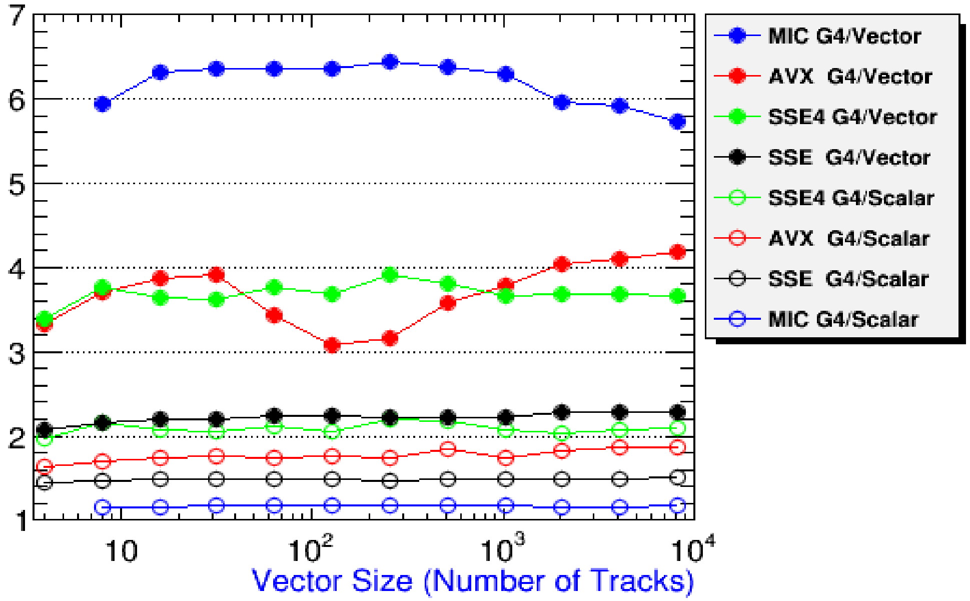
- **Gains of up to x2 in single thread mode with no vectorisation need to be fully understood, not explained only by:**
  - ~x5 less instruction cache misses
  - ~20% less CPI



VECTORIZING PHYSICS

- Physics vectorisation has started with the electromagnetic processes
- The vectorised Compton scattering shows good performance gains
- Vector code is better scalar code!
- We will consider to retrofit this into Geant4

Compton benchmark



ROAD PATH

- **Vectorized version of CMS2015 benchmark on Xeon**
  - VecGeom global navigation in CMS
  - Fall 2015
- **Gradual addition of vector physics models in the prototype workflow**
  - EM could have good progress by the end of 2015
  - Requires interfaces (partly done), models, basketizing on physics
- **KNC/KNL and GPU benchmarks**
  - Use as coprocessor/native mode
  - KNC&GPU in offload mode by the fall
- **Improve scalability and enable MIMD mode**
  - Dispatch multiprocessor and multi node
- **Integrate more realistic setups (more LHC experiments)**
  - Including scoring/digitization/I/O
- **I/O of kinematics, support for user hits/digits**
  - Offer a choice of a factory based, generic multithreaded I/O
- **0.0 release aimed for end of the year**
  - User able to simulate in realistic geometry using tabulated physics
- **Alpha release for 2018**
  - Usable at large scale, most features available